AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions and listings of claims in the application:

LISTING OF CLAIMS:

A1

1. (currently amended): A data packet classifier to classify a plurality of N-bit input Internet Protocol network tuples, wherein N is larger than or equal to 84, said classifier comprising:

a hash address generator to generate a plurality of M-bit hash addresses from said plurality of N-bit input tuples, wherein M is significantly smaller than N and M at least smaller than 32;

a memory having a plurality of memory entries, said memory being addressable by said plurality of M-bit hash addresses, each such address corresponding to a plurality of memory entries, each of said plurality of memory entries capable of storing one of said plurality of N-bit tuples and an associated process flow information;

a comparison unit to determine if an incoming N-bit tuple can be matched with a stored N-bit tuple, wherein said associated process flow information is output if a match is found and wherein a new entry is created in the memory for the incoming N-bit tuple if a match is not found,

wherein said N-bit tuple includes information about a source address, a destination address, protocol, a source port and a destination port.

2. (original): The data packet classifier of claim 1 further comprising:

a content addressable memory (CAM) to store overflowing N-bit tuples and their corresponding flow information wherein said overflowing N-bit tuple cannot be stored in the memory.

3. (original): The data packet classifier of claim 1 wherein said process flow information in the memory comprises a flow identification number.

A1

4. (original): The data packet classifier of claim 1 wherein said process flow information in the memory can be updated.

5. (original): The data packet classifier of claim 1 wherein an entry in the memory can be deleted.

6. (original): The data packet classifier of claim 1 wherein searching for an entry in the memory can be ceased when a kill-process command is received.

7. (original): The data packet classifier of claim 2 wherein said process flow information in the CAM comprises a process flow identification number.

8. (original): The data packet classifier of claim 2 wherein said process flow information in the CAM can be updated.

9. (original): The data packet classifier of claim 2 wherein an entry in the CAM can be deleted.

10. (original): The data packet classifier of claim 2 wherein searching for an entry in the CAM can be ceased when a kill-process command is received.

A1

11. (original): The data packet classifier of claim 2 further capable of generating a trap if both the memory and the CAM are full.

12. (original): The data packet classifier of claim 2 wherein both the memory and CAM are searched in parallel.

13. (original): The data packet classifier of claim 2 wherein N > 96.

14. (original): The data packet classifier of claim 13 wherein said hash address generator performs hashing on a first 96 bits of an associated N-bit tuple.

15. (original): The data packet classifier of claim 14 wherein a comparison of tuple stored in the memory and an incoming tuple is performed using three 32-bit comparators and standard 16 or 32 bit wide memories.

16. (currently amended): A network system comprising a plurality of nodes, each of said nodes having a unique N-bit Internet Protocol network tuple, wherein N is larger than or equal to 84, each of said plurality of nodes comprising a data packet classifier, said data packet classifier comprising:

a hash address generator to generate a plurality of M-bit hash addresses from said plurality of N-bit input tuples, wherein M is significantly smaller than N and M at least smaller than 32;

A1

a memory having a plurality of memory entries, said memory being addressable by said plurality of M-bit hash addresses, each such address corresponding to a plurality of memory entries, each of said plurality of memory entries capable of storing one of said plurality of N-bit tuples and an associated process flow information;

a comparison unit to determine if an incoming N-bit tuple can be matched with a stored N-bit tuple, wherein said associated process flow information is output if a match is found and wherein a new entry is created in the memory for the incoming N-bit tuple if a match is not found,

wherein said N-bit tuple includes information about a source address, a destination address, protocol, a source port and a destination port.

17. (original): The data packet classifier of claim 16 further comprising:

a content addressable memory (CAM) to store overflowing N-bit tuples and their corresponding process flow information wherein said overflowing N-bit tuple cannot be stored in the memory.

18. (original): The data packet classifier of claim 16 wherein said process flow information in the memory comprises a process flow identification number.

19. (original): The data packet classifier of claim 16 wherein said process flow information in the memory can be updated.

A1

20. (original): The data packet classifier of claim 16 wherein an entry in the memory can be deleted.

21. (original): The data packet classifier of claim 16 wherein searching for an entry in the memory can be ceased when a kill-process command is received.

22. (original): The data packet classifier of claim 17 wherein said process flow information in the CAM comprises a process flow identification number.

23. (original): The data packet classifier of claim 17 wherein said process flow information in the CAM can be updated.

24. (original): The data packet classifier of claim 17 wherein an entry in the CAM can be deleted.

25. (original): The data packet classifier of claim 17 wherein searching for an entry in the CAM can be ceased when a kill-process command is received.

26. (original): The data packet classifier of claim 17 further capable of generating a trap if both the memory and the CAM are full.

26. (original): The data packet classifier of claim 17 wherein both the memory and CAM are searched in parallel.

A1

28. (original): The data packet classifier of claim 17 wherein N > 96.

29. (original): The data packet classifier of claim 17 wherein said hash address generator performs hashing on a first 96 bits of an associated N-bit tuple.

30. (original): The data packet classifier of claim 29 wherein a comparison of tuple stored in the memory and an incoming tuple is performed using three 32-bit comparators.

31. (currently amended): A method of generating an M-bit hash address from an N-bit input tuple comprising:

a) splitting said N-bit input tuple into a first range of X bits and a second range of Y bits, where X is equal to or smaller than M, wherein N is larger than or equal to 84 and M is at least smaller than 32;

b) applying a hash function to said X bits to generate a white hash address with Z bits, where Z is equal to or smaller than M;

c) creating said M-bit hash address by combining said Z-bit white hash address and said second range of Y bits using a Boolean operator,

wherein said N-bit tuple includes information about a source address, a destination address, protocol, a source port and a destination port related to a packet in a network.

32. (currently amended): The method of claim 31 wherein X is significantly larger than Y and X is at least two thirds of N.

A\

33. (currently amended): The method of claim 31 wherein X is significantly larger than Z and Z is smaller than 32.

34. (original): The method of claim 31 wherein said Boolean operator is an OR.

35. (original): The method of claim 31 wherein said Boolean operator is an XOR.

36. (original): The method of claim 31 wherein N is 104.

37. (original): The method of claim 36 wherein X is 96 and Y is 8.

38. (original): The method of claim 37 wherein Z is 20 and M is 20.

39. (currently amended): A computer program product, including a computer-readable medium comprising instructions, said instructions enabling a computer to perform a hashing function on an N-bit input tuple according to the following steps:

a) splitting said N-bit input tuple into a first range of X bits and a second range of Y bits, wherein N is larger than or equal to 84 and M is at least smaller than 32;

b) applying a hash function to said X bits to generate a white hash address with Z bits;

c) creating said M-bit hash address by combining said Z-bit white hash address and said second range of Y bits using a Boolean operator

A1

wherein said N-bit tuple includes information about a source address, a destination address, protocol, a source port and a destination port related to a packet in a network.

40. (currently amended): The program product of claim 39 wherein X is significantly larger than Y and X is at least two thirds of N.

41. (currently amended): The program product of claim 39 wherein X is significantly larger than Z and Z is smaller than 32.

42. (original): The program product of claim 39 wherein said Boolean operator is an OR.

43. (original): The program product of claim 39 wherein said Boolean operator is an XOR.

44. (original): The program product of claim 39 wherein N is 104.

45. (original): The program product of claim 44 wherein X is 96 and Y is 8.

46. (original): The program product of claim 45 wherein Z is 20 and M is 20.

A1

47. (currently amended): A computer program product, including a computer-readable medium comprising instructions, said instructions comprising:

a hash address generator code to enable a computer to generate a plurality of M-bit hash addresses from said plurality of N-bit input Internet Protocol network tuples, wherein N is larger than or equal to 84, and wherein M is significantly smaller than N and M at least smaller than 32;

a memory code to enable a computer to store data in a memory having a plurality of memory entries, said memory code further enabling the computer to address said plurality of M-bit hash addresses, each of said plurality of memory entries capable of storing one of said plurality of N-bit tuples and an associated process flow information;

a comparison code to determine if an incoming N-bit tuple can be matched with a stored N-bit tuple, wherein said associated process flow information is output if a match is found and wherein a new entry is created in the memory for the incoming N-bit tuple if a match is not found.

48. (original): The computer program product of claim 47 further comprising:

a content addressable memory (CAM) code to enable a computer to store overflowing N-bit tuples and their corresponding process flow information in a CAM wherein said overflowing N-bit tuple cannot be stored in the memory.

49. (original): The computer program code of claim 47 wherein said process flow information in the memory comprises a flow identification number.

A1

50. (original): The computer program code of claim 47 wherein said instructions enable a computer to update the process flow information.

51. (original): The computer program code of claim 47 wherein said instructions enable a computer to delete an entry in the memory.

52. (original): The computer program code of claim 47 wherein said instructions enable a computer to cease searching for an entry in the memory when a kill-process command is received.

53. (original): The computer program code of claim 48 wherein said process flow information in the CAM comprises a flow identification number.

54. (original): The computer program code of claim 48 wherein said instructions enable a computer to update the process flow information in the CAM.

55. (original): The computer program code of claim 48 wherein said instructions enable a computer to delete an entry in the CAM.

56. (original): The computer program code of claim 48 wherein said instructions enable a computer to cease searching for an entry in the CAM when a kill-process command is received.

A1

57. (original): The computer program product of claim 48 wherein said instructions enable the computer to generate a trap if both the memory and the CAM are full.

58. (original): The computer program product of claim 48 wherein said instructions enable a computer to search both the memory and CAM in parallel.

59. (original): The computer program product of claim 48 wherein N > 96.

60. (original): The computer program product of claim 59 wherein said hash address generator code enables a computer to performs hashing on a first 96 bits of an associated N-bit tuple.

61. (original): The computer program product of claim 60 wherein said comparison code enables the computer to compare an address stored in the memory and an incoming tuple using three 32-bit comparators.

A\ 62. (original): The method of claim 31 wherein said Boolean operator is an “AND”.

63. (original): The computer program product of claim 39 wherein said Boolean operator is an “AND”.